Transformation Logic for the Role Pattern from PURO to OntoUML

Tomáš Březina[1]

[1] *Faculty of Informatics and Statistics, Prague University of Economics and Business, Prague, Czechia*

Abstract

OntoUML is a conceptual modeling language based on the Unified Foundational Ontology (UFO), yet its high expressivity often presents a steep learning curve for novice modelers. The PURO language offers a lightweight sketching alternative that allows users to articulate domain situations at the instance level before transitioning to formal representations. Building upon the baseline PURO-to-OntoUML transformation framework, this paper proposes a method to address current limitations in Role pattern transformation by transitioning from 1:1 mapping toward contextual pattern matching and automated identity deduction. The proposed approach combines contextual pattern matching, rule-based automated identity deduction, and consistency verification to reduce manual intervention while preserving compliance with UFO-A constraints. Using a running domain example, we illustrate how these mechanisms resolve Role-specific ambiguities and streamline transformation from lightweight PURO sketches to formal OntoUML models.

Keywords

Ontology Engineering, OntoUML, UFO, PURO, Conceptual Modeling, Role Pattern

1. Introduction

A persistent challenge in ontology engineering is the division between the semantic web community, emphasizing OWL [1] and RDF [2], and the applied ontology community, prioritizing deep foundational distinctions. Although knowledge engineering heavily favors OWL, ontologies derived from OntoUML [3] and grounded in the UFO [3] help improve formal correctness and reduce semantic misinterpretation. Furthermore, the gUFO library [4] enables straightforward transformation from OntoUML into standard OWL. However, authoring directly in OntoUML presents a steep entry barrier. Therefore, PURO was conceptualized as a graphical sketching approach for instance-level domain situations. Initial empirical analyses indicate that this "sketch-first" paradigm improves domain comprehension compared to direct formal authoring [5]. While originally designed to generate OWL ontologies directly [6], PURO was later adapted as a front-end sketching tool for OntoUML, establishing a two-step workflow: first sketching in PURO, then transforming the sketch into OntoUML (and subsequently OWL). This paper focuses on transforming the Role pattern in the first step of this pipeline.

Traditional tools like OLED or Menthor rely on strict model verification to ensure compliance with UFO axioms [7, 8], whereas PURO offers an alternative visual approach

The 25th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2026) - Satellite Events (Workshops, Tutorials, Demos, Posters)

bret04@vse.cz (T. Březina)

0009–0004–4075–9262 (T. Březina)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-4340/paper13.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-26

intended to lower initial modeling barriers [6]. Based on these paradigms, a prototype framework established a dialogue-based bridge between PURO sketches and OntoUML class diagrams [6]. Despite these advantages, existing tools still require substantial user intervention, particularly for context-dependent relational patterns. Building on prior work [6], this paper makes three contributions. First, it introduces context-aware pattern matching for identifying Role-related structures that cannot be resolved through isolated one-to-one mapping. Second, it proposes automated identity provider deduction based on structural and instance-level evidence. Third, it extends the transformation workflow with relator injection and consistency verification to ensure compliance with UFO-A constraints. Together, these mechanisms reduce manual intervention while addressing ternary structure blindness and identity ambiguity.

2. PURO and OntoUML

PURO and OntoUML operate at different abstraction levels, requiring a mapping between instance-level situations and universal-level conceptual categories. In UFO-A, these categories distinguish between sortals and non-sortals. Sortals are types that carry or supply a principle of identity for their instances, such as a Kind or Role, whereas non-sortals represent shared properties across different identity providers without supplying identity, such as a Category.

2.1. PURO

PURO is based on foundational distinctions between objects and relationships, as well as particulars and universals, which serve as its core modeling concepts [5]. The language employs a small set of modeling primitives that can be combined flexibly without the premature decidability constraints imposed by OWL [6]. These primitives are divided into particulars, representing singular entities that cannot have instances (B-Object, B-Relationship, and B-Valuation), and universals, representing general categories that can be instantiated (B-Type, B-Relation, and B-Attribute). Figure 1a shows a PURO model from the cooking recipe domain, depicting instances such as a specific BoiledEgg playing the role of an ingredient in a dish.

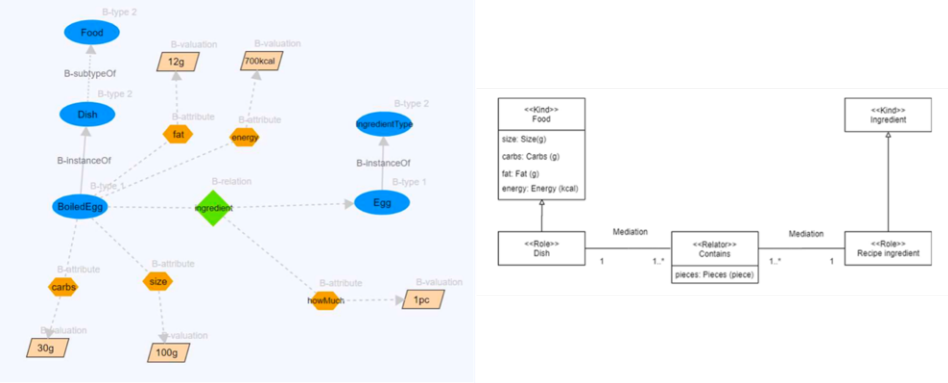


Figure 1: (a) PURO model in the cooking recipe domain; (b) Target OntoUML model [author].

2.2. OntoUML

OntoUML extends standard UML class diagrams with stereotypes derived from UFO-A, the structural layer of the UFO. Central to OntoUML is the distinction between rigid types (e.g.,

Kind), which classify instances necessarily, and anti-rigid types (e.g., Role), which represent contingent properties [3].

In OntoUML, a Role is an anti-rigid specialization of an identity provider that exists only within a relational context. Unlike standard UML and OWL, which typically represent relations as binary, UFO-A models material relations through an underlying n-ary structure mediated by a Relator. The Relator mediates multiple existentially dependent Roles. As shown in the cooking recipe model in Figure 1b, the Contains Relator mediates between Dish and Recipe ingredient Roles. Roles are governed by two essential meta-model constraints [3]: Constraint C1 (Identity Provision): A Role must always have exactly one identity provider (e.g., Kind, Collective) as a direct or indirect supertype; Constraint C2 (Relational Dependency): Every Role must be connected via a Mediation relation to a Relator, as it cannot exist outside of its specific relational context. These constraints motivate the transformation logic proposed in this paper.

3. Baseline Transformation Framework

The current state of PURO-to-OntoUML transformation is defined by a prototype framework[2]. This framework utilizes an interactive, dialogue-based approach to guide users through the modeling process, acting as a bridge between lightweight instance sketching and formal OntoUML structures [6]. When the system encounters ambiguous elements, it presents dialogue prompts while incrementally constructing the OntoUML model (Figure 2).

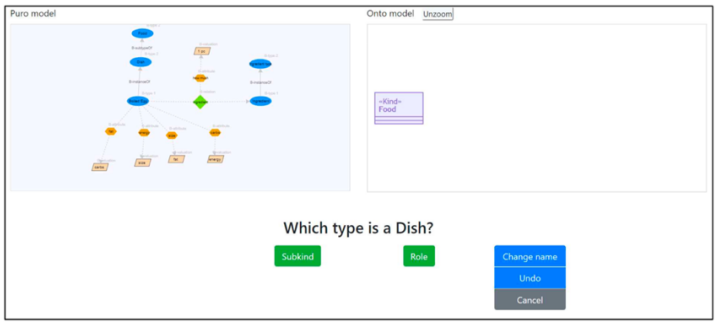


Figure 2: Manual type disambiguation dialogue for Dish in the baseline tool [author].

Although suitable for basic 1:1 transformations, technical analysis reveals key limitations when processing complex patterns such as Roles, as illustrated by our running cooking recipe example (Figures 1-2): **(1) Manual Type Disambiguation:** In the PURO sketch (Figure 1a), Dish is modeled as a B-Type extending Food. Because the tool evaluates nodes in isolation using 1:1 rules, it cannot determine whether Dish is a Subkind or a Role, forcing a manual user prompt asking "Which type is a Dish?" (Figure 2); **(2) Binary Bias and Relator Blindness:** Modelers frequently represent relations directly between entities (e.g., connecting BoiledEgg to Egg via an ingredient link) without explicit Relators. The baseline tool's binary graph mapping logic fails to detect that this relationship requires an OntoUML Relator (Contains), failing to generate the required Mediation relations and Relator class shown in Figure 1b; **(3) Loss of Identity hierarchy:** In PURO, connections between contingent types and fundamental categories are

[2] Available at https://github.com/marek-dudas/puro-onto

often sketched as informal associations. As a result, the tool fails to automatically enforce Constraint C1, requiring manual intervention to assert that Recipe ingredient derives its identity from the Ingredient Kind (Figure 1b).

4. Proposed Method for Role Transformation

To address the baseline limitations, we propose an updated method moving from 1:1 mapping toward contextual pattern matching and automated identity deduction. This method analyzes relational context using structural analysis and instance-level heuristics to satisfy UFO-A constraints, while reducing manual dialogue prompts. Unlike the baseline framework, which evaluates each PURO node independently, our approach reasons over the local graph structure to infer OntoUML semantics before user interaction. The novelty lies in using contextual semantic inference to resolve Role-specific ambiguities during PURO-to-OntoUML transformation. We realize this approach through three complementary mechanisms.

4.1. Contextual Pattern Matching

The proposed transformation evaluates a relation node together with its neighborhood. If a central relationship node R in the PURO sketch connects two or more participating entities or includes valuations, the logic prioritizes its transformation into an OntoUML Relator. Formally, if a B-Relation R is associated with participating B-Types or B-Objects $T_1, T_{2,\dots,} T_n$ $(n \geq 2)$, the mapping assigns R to a Relator stereotype:

$$\text{CPM}(R) \Rightarrow \text{stereotype}(R,\text{Relator}) \tag{1}$$

For each participating T_i (where $i \in \{1, \dots, n\}$), the relationship is mapped to an OntoUML Mediation relation M_i connecting T_i to the newly designated Relator.

$$\forall T_i \in \{T_1, \dots, T_n\}, \;\; \text{stereotype}(M_i,\text{Mediation}) \wedge \text{connects}(M_i, T_i, R) \tag{2}$$

In the cooking recipe example introduced in Figure 1a, the baseline framework treated the ingredient B-Relation (Figure 1a) as a simple binary link, missing the necessary Relator structure. Under Equations (1) and (2), the engine evaluates ingredient along with its connected valuations (howMuch). It automatically transforms ingredient into the Contains Relator shown in Figure 1b and generates the corresponding Mediation associations connecting Dish and Recipe ingredient, satisfying Constraint C2 (Relational Dependency). This avoids manual Relator creation when sufficient structural evidence is present.

4.2. Automated Identity Provider Deduction

To satisfy Constraint C1, the method employs two deductive mechanisms.

Structural Analysis: The rules controller traverses paths in the PURO B-Type hierarchy. If a subsumption relationship exists between two entities T_1 and T_2 $(T_2 \sqsubseteq_B T_1)$ and T_2 is relationally dependent, the algorithm infers that T_1 is the identity-providing Kind and T_2 is a specialized Role:

$$(T_2 \sqsubseteq_B T_1) \wedge \mathit{dependent}(T_2) \Rightarrow \text{stereotype}(T_2,\text{Role}) \wedge \text{stereotype}(T_1,\text{Kind}) \wedge (T_2 \sqsubseteq_{OU} T_1) \tag{3}$$

In Figure 1a, Dish is modeled as a subtype of Food ($Dish \sqsubseteq_B Food$). Rather than triggering the manual user prompt shown in Figure 2 ("Which type is a Dish?"), Equation (3) detects that Dish participates in the ingredient relational context and automatically assigns Food as the Kind and Dish as the specialized Role (Figure 1b).

Instance-Level Analysis: When structural hierarchies are absent, the system analyzes instance assertions. If an individual (i) belongs to both a base category T_1 and a relationally active B-Type T_2, the engine applies a non-monotonic heuristic inference:

$$(i:T_1) \wedge (i:T_2) \wedge \text{active}(T_2) \Longrightarrow \text{stereotype}(T_1,\text{Kind}) \wedge \text{stereotype}(T_2,\text{Role}) \wedge (T_2 \sqsubseteq_{OU} T_1) \quad (4)$$

For example, the instance BoiledEgg (i) in Figure 1a is typed as both Food (T_1) and Dish (T_2). Because Dish actively connects to the ingredient relation while Food has no external relational dependency, Equation (4) heuristically deduces that Food provides the identity for Dish, presenting this to the user as a pre-filled suggestion rather than an unguided question.

4.3. Relator Injection and Consistency Verification

If a PURO sketch defines a Role (such as Recipe ingredient in Figure 1b) but omits an explicit relational universal, the event manager performs a relator injection, inserting an anonymous Relator into the virtual model to anchor Mediation associations. Finally, a verification routine checks the model against UFO-A axioms to ensure every Role is bound to an identity provider (Constraint C1) and a mediating Relator (Constraint C2) before closing the transformation branch. Overall, these mechanisms reduce manual modeling decisions during transformation.

5. Future Directions: AI-Driven Transformation

While the proposed rule-based logic relies on graph structure and instance overlap, integrating Large Language Models (LLMs) offers a complementary pathway. Future iterations could leverage LLMs for zero-shot linguistic classification to infer relational dependence in concept labels prior to graph traversal. Additionally, multi-agent workflows [9] could act as automated validation oracles, evaluating candidate models against Constraints C1 and C2 and providing natural language justifications during user dialogue resolution [10].

6. Conclusions

This paper proposes an updated transformation logic for role modeling in the PURO-to-OntoUML pipeline. By transitioning from simple 1:1 node mappings to contextual pattern matching and automated identity deduction, the proposed approach helps reduce baseline limitations, such as relator blindness and repetitive manual type prompts. Overall, the proposed transformation logic shifts the PURO-to-OntoUML workflow from isolated rule-based mapping to context-aware semantic inference, reducing manual modeling effort for ontology engineers.

Declaration on Generative AI

During the preparation of this work, the author used Gemini to: Grammar and spelling check, paraphrase, and reword.

References

[1] W3C OWL Working Group. (2009). OWL 2 Web Ontology Language Document Overview. W3C Recommendation, World Wide Web Consortium.

[2] Cyganiak, R., Wood, D., & Lanthaler, M. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, World Wide Web Consortium.

[3] Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Models. Telematica Instituut Fundamental Research Series, Enschede.

[4] Almeida, J. P. A., Guerson, J., Guizzardi, G., Sales, T. P., & Fonseca, C. M. (2019). gUFO: A Lightweight Implementation of the Unified Foundational Ontology (UFO) for the Semantic Web. In: Proceedings of the Joint Ontology Workshops (JOWO 2019), CEUR-WS, Vol. 2518.

[5] Dudáš, M., Svátek, V., Vacura, M., & Zamazal, O. (2016). Starting Ontology Development by Visually Modeling an Example Situation – a User Study. In: EKAW 2016 Posters and Demos, CEUR Workshop Proceedings, Vol. 1704.

[6] Dudáš, M., Morkus, T., Svátek, V., et al. (2020). Kickstarting OntoUML Modeling from PURO Instance-Level Examples. In: EKAW Posters and Demos 2020, CEUR Workshop Proceedings, Vol. 2751, pp. 36–40.

[7] Guerson, J., Sales, T. P., Guizzardi, G., & Almeida, J. P. A. (2015). OntoUML Lightweight Editor: A Model-Based Environment to Build, Evaluate and Implement Reference Ontologies. In: 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop (EDOCW), IEEE, pp. 144–153.

[8] Mykoliuk, A. (2026). Comparative analysis of tools supporting the design of OntoUML models. Bachelor's thesis, Prague University of Economics and Business.

[9] Salovskii, P., & Gorshkova, I. (2026). Automatic Ontology Construction Using LLMs as an External Layer of Memory, Verification, and Planning for Hybrid Intelligent Systems. arXiv preprint arXiv:2604.20795.

[10] Lushnei, S., et al. (2026). Large Language Models as Oracles for Ontology Alignment. In: Proceedings of EACL 2026.